Improve Fidelity and Utility of Synthetic Credit Card Transaction Time Series from Data-centric Perspective

Din-Yin Hsieh*, Chi-Hua Wang †, Guang Cheng‡

January 3, 2024

Abstract

Exploring generative model training for synthetic tabular data, specifically in sequential contexts such as credit card transaction data, presents significant challenges. This paper addresses these challenges, focusing on attaining both high fidelity to actual data and optimal utility for machine learning tasks. We introduce five pre-processing schemas to enhance the training of the Conditional Probabilistic Auto-Regressive Model (CPAR), demonstrating incremental improvements in the synthetic data's fidelity and utility. Upon achieving satisfactory fidelity levels, our attention shifts to training fraud detection models tailored for time-series data, evaluating the utility of the synthetic data. Our findings offer valuable insights and practical guidelines for synthetic data practitioners in the finance sector, transitioning from real to synthetic datasets for training purposes, and illuminating broader methodologies for synthesizing credit card transaction time series.

Key Words: Credit Card Transaction Data, Time Series Generative Model, Synthetic Training Datasets, Fraud Detection, Data-centric Machine Learning..

1 Introduction

Synthetic Tabular Data is garnering increasing interest from the academic and industrial sectors alike [13]. This surge in attention can be attributed to the capability of synthetic data to foster innovation and aid in decision-making processes, all while adhering to contemporary privacy regulations such as GDPR [10] and CCPA [27]. The promising prospects of accelerating and supporting the entire machine learning lifecycle make this area particularly

*Undergraduate Student, Department of Statistics and Data Science, UCLA, CA, 90095. Email: darrenhsieh1205@g.ucla.edu

†Postdoctoral Scholar, Department of Statistics and Data Science, UCLA, CA, 90095. Email: chihuawang@ucla.edu

‡Professor, Department of Statistics and Data Science, UCLA, CA, 90095. Email: guangcheng@ucla.edu

appealing. As a result, there is a growing impetus among machine learning researchers and practitioners to delve into the advantages and limitations associated with utilizing synthetic datasets [29, 9].

While synthetic tabular data offers substantial potential, its application is fraught with challenges, particularly in handling sequential tabular data [33]. Despite its apparent similarity to standard tabular data in terms of representation, sequential tabular data is unique due to the existence of *relationships between rows*. These inter-row relationships arise from the specific nature of the application used during data collection. Consider, for example, health monitoring data: here, data points (or rows) are recorded at consistent time intervals, resulting in a regular and predictable pattern conducive to statistical modeling [18, 32]. In contrast, credit card transaction data presents a different scenario; the time intervals between data points (transactions) are irregular, mirroring the sporadic nature of consumer spending. Consequently, the regularity of data point recording emerges as a crucial consideration in the development of models for synthetic sequential data.

In this study, we aim to offer insights on training generative models tailored for the *credit card transaction data synthesis*, ensuring both high fidelity to real data and optimal utility for machine learning applications. It's important to note that credit card transaction data is inherently multi-sequence, marked by its irregular measurement intervals [30]. Presently, the CPAR (Conditional Probabilistic Auto-Regressive Model) [33] stands out in literature as a quintessential multi-sequence generative model. CPAR model is available to researchers and practitioners through the open-source Synthetic Data Vault (SDV) library [21] (for a detailed understanding of the CPAR model, please refer to Section 3.2). Although the CPAR model can handle both regular and irregular multi-sequence data, its synthetic output for credit card transaction data still leaves room for enhancement in terms of real data fidelity and machine learning utility. Our experiments shed light on the criticality of "pre-processing" in tabular data fields, which markedly influences the fidelity and utility of the synthesized credit card transactions. Such discoveries emphasize the importance of thorough data preparation to elevate the efficacy of generative models, especially when handling intricate datasets like credit card transaction records.

We adopt a *data-centric* strategy to bolster the fidelity and utility of synthetic credit card transaction data. Instead of tweaking the generative model to enhance the quality of synthetic data, we place our primary emphasis on the *preprocessing of raw tabular data*. This shift in focus aims to amplify both the fidelity and utility of the derived synthetic time series. We propose five unique preprocessing schemas, each designed to iteratively enhance the fidelity of the generated synthetic credit card data. After attaining a satisfactory fidelity benchmark, we pivot our efforts toward training fraud detection models [5, 25, 19] specifically tailored for time-series data. Three models — `XGBoost` [4], `LGBM` [14], and `Catboost` [8] — were chosen and trained on data stemming from the most promising preprocessing schema. Our exploration centers around the efficacy of these models, with a special focus on their False Positive Rate (FPR) and False Negative Rate (FNR) in the fraud detection context (for a detailed analysis, please see Section 4.3). Through this comprehensive approach, we accentuate not just the pivotal role of preprocessing in the realm of synthetic time series data generation, but also its broader ramifications for ensuing machine learning tasks, laying a solid groundwork for ensuing investigations in this area.

Paper Organization. This paper is structured as follows. Section 2 gives a review on

related keywords about synthetic credit card transaction data. Section 3 give comprehensive details on how to preprocess the credit card dataset to train CPAR model for high-fidelity and high-utility. Section 4 gives results on the fidelity evaluation of categorical variables (Section 4.1) and continuous variable (Section 4.2) and also utility evaluation of fraud detection model performance (Section 4.3). Section 5 talks about our conclusion, new concerns and future direction.

2 Relate Work

2.1 Synthetic Transactions Dataset

In recent years, the generation of synthetic data has emerged as a popular research direction [2, 22, 28], primarily due to the increasing accessibility and ease of use of various tabular synthesis models. These models encompass methodologies based on Generative Adversarial Networks (GANs) [31, 36, 35, 17, 34], Diffusion Models [16, 15], as well as Transformers [12, 3, 26], all of which have achieved success across diverse domains of tabular data. Nevertheless, the synthesis of transactional data poses unique research challenges. Transactional data inherently constitutes a time series, and the successful applications of synthetic data models have predominantly been on tabular data lacking temporal stamps. This discrepancy highlights the complexity and specificity required in handling time-series transactional data, necessitating further investigation and innovation in this domain.

2.2 CPAR model

The *Conditional Probabilistic Auto-Regressive*, or the CPAR model [33] is the main model that aims to capture sequential dependencies currently provided by the *Synthetic Data Vault* (SDV) [21]. The model, which is based on neural network, achieves this by capturing inter-row dependencies conditioned on the previous sequence history, and then outputting the necessary parameters to synthesize the future entries. For training, this model also has 3 different loss functions, with each applied to 3 different types of data: continuous numerical, discrete numerical, and categorical.

2.3 Credit Card Transaction Dataset

We use credit card transaction dataset provided in [2]. According to the offical documentation of *Synthetic Data Vault* (SDV), the CPAR model is suitable for multi-sequence data, which fits our credit card transaction dataset where there exists multiple users, each with their own transaction history. In addition, we were curious about CPAR's ability to generate high fidelity synthetic data given columns that have continuous numerical data, as well as columns that have high cardinality. Furthermore, the existence of the `Is Fraud?` column gave us the ability to generate synthetic data to train machine learning models for downstream tasks such as fraud detection. Overall, we believe that this dataset is great for both assessing the quality of the CPAR model, as well as the machine learning efficacy of synthetic dataset for downstream tasks.

Part 1 of Full Credit Card Transaction Dataset

	Categorical	Categorical	Categorical	Numerical (continuous)	Categorical	Categorical
	User	Card	Use Chip	Amount	MCC	Errors?
$s_0^{(1)}$	214	0	Swipe Transaction	$3.94	5411	NaN
$s_1^{(1)}$	214	0	Swipe Transaction	$28.69	5411	NaN
$s_2^{(1)}$	214	0	Swipe Transaction	$7.05	5541	NaN
$s_t^{(1)}$	...	...	...	...	...	...
$s_0^{(2)}$	882	0	Chip Transaction	$53.87	5541	NaN
$s_1^{(2)}$	882	0	Chip Transaction	$82.00	5541	NaN
$s_2^{(2)}$	882	0	Chip Transaction	-$82.00	5541	NaN
$s_t^{(2)}$	...	...	...	...	...	...

Part 2 of Full Credit Card Transaction Dataset

	Datetime	Datetime	Datetime	Datetime
	Year	Month	Day	Time (HH:MM)
$s_0^{(1)}$	2018	1	1	06:05
$s_1^{(1)}$	2018	1	1	13:08
$s_2^{(1)}$	2018	1	1	13:20
$s_t^{(1)}$	...	...	...	...
$s_0^{(2)}$	2016	4	2	12:02
$s_1^{(2)}$	2016	4	2	12:09
$s_2^{(2)}$	2016	4	2	12:15
$s_t^{(2)}$	...	...	...	...

Part 3 of Full Credit Card Transaction Dataset

	Categorical	Categorical	Categorical	Categorical
	Merchant Name	Merchant City	Merchant State	Zip
$s_0^{(1)}$	-8125167349407750106	Saint Petersburg	FL	33706.0
$s_1^{(1)}$	-8125167349407750106	Saint Petersburg	FL	33706.0
$s_2^{(1)}$	2027553650310142703	Saint Petersburg	FL	33707.0
$s_t^{(1)}$	...	...	...	...
$s_0^{(2)}$	2027553650310142704	Sherwood	AR	72120.0
$s_1^{(2)}$	2027553650310142705	Sherwood	AR	72120.1
$s_2^{(2)}$	2027553650310142706	Sherwood	AR	72120.2
$s_t^{(2)}$	...	...	...	...

Figure 1: Metadata Data Type of Original Credit Card Transaction Dataset. $S_j^{(i)}$ denotes the ith user's jth row.

3 Approach and Evaluation Framework

This section gives comprehensive details on the credit card transaction dataset (Sec. 3.1), training CPAR model (Sec. 3.2), how to preprocess the credit card transaction dataset (Sec. 3.3) and how to train the fraud detection model with synthetic data (Sec. 3.4).

3.1 Basic of Credit Card Transaction Dataset

Figure 1 gives the full table of the Credit Card Transaction Dataset. The figure, containing 3 parts, shows an overview of the original dataset and its columns, as well as what types of data they are considered under the CPAR framework. In part 1, the figure shows that for columns `User`, `Card`, `Use Chip`, `MCC`, and `Errors?`, the CPAR framework considers them as categorical data columns. The `Amount` column, on the other hand, is considered as a continuous numerical data column.

In part 2, the figure shows that the columns `Year`, `Month`, `Day`, and `Time` are not considered as any of the 3 data types (continuous numerical, discrete numerical, or categorical). Rather, they are considered as 'datetime', which are used to set the sequence index under the metadata of the CPAR model.

In part 3, the figure shows that the columns `Merchant Name`, `Merchant City`, `Merchant State`, and `Zip` are considered as categorical data columns. Note this is still the case when data in `Merchant Name` are of type 'integer', as the original dataset follows the same form for this column. This also applies for the `Zip` column, where all zip codes are displayed as

'float'. In the synthesis of the original data (except for schema 1), we transform the data in these 4 columns into 'string' and consider them as categorical columns when training the CPAR model.

3.2 CPAR Basic

To train the CPAR model on a dataset of credit card transactions, we categorize the loss functions according to the data type of the variables involved: continuous numerical and categorical.

Continuous Numerical Data: Consider the parameters $\mu = \pi^{(i)}_{(t,\mu)}$, $\sigma = \pi^{(i)}_{(t,\sigma)}$, and $m = \pi^{(i)}_{(t,m)}$, where i represents the sequence number (or user), and t denotes the transaction index for a given user. The loss function $L(x; \mu, \sigma, m)$ is defined as: $\mathcal{L}(x; \mu, \sigma, m) = -\left(\log\left(f_{\mu,\sigma^2}(x)\right) + \log(1-m)\right)$: x is not missing; $\mathcal{L}(x; \mu, \sigma, m) = -\log(m)$: x is missing. Here, μ and σ^2 represent the mean and variance of a Gaussian distribution, and m indicates the probability of the value being missing. In our case, the credit card transaction dataset contains a continuous numerical column, `Amount`, parameterized by: $\pi^{(i)}_{(t,\mu)}, \pi^{(i)}_{(t,\sigma)}, \pi^{(i)}_{(t,m)}$ where $i \in \{0, 1, 2\}$, $t \in \{2540, 2676, 2630\}$, and $m \in \{0, 0, 0\}$ corresponding to users 214, 882, and 1798 respectively.

Categorical Data:For categorical data, the loss function is defined as:

$$L(x; \pi_0, \pi_1, \ldots, \pi_{N-1}) = -\sum_{j \in N} x_j \log(\pi_j)$$

where N is the number of categories, π_j represents the proportion of category j in the dataset, and x_j is a binary indicator, equal to 1 if the instance belongs to category j and 0 otherwise. In our dataset, the following columns are treated as categorical: `Card`, `Use Chip`, `Merchant Name`, `Merchant City`, `Merchant State`, `Zip`, `MCC`, `Errors?`, and `Is Fraud?`. Their parameters are denoted as $\pi^{(i)}_{(t,j)}$, with the index sets being: $i \in \{0, 1, 2\}, t \in \{2540, 2676, 2630\}, j = 0$ and the category counts, N, detailed as follows:

1. `Card`: $N = 1$,
2. `Use Chip`: $N \in \{2, 3, 3\}$,
3. `Merchant Name`: $N \in \{185, 151, 231\}$,
4. `Merchant City`: $N \in \{68, 78, 115\}$,
5. `Merchant State`: $N \in \{20, 23, 29\}$ (excluding NaN),
6. `Zip`: $N \in \{87, 86, 134\}$ (excluding NaN),
7. `MCC`: $N \in \{63, 62, 73\}$,
8. `Errors?`: $N \in \{5, 4, 2\}$ (excluding NaN),
9. `Is Fraud?`: $N = 2$.

Algorithm 1 One training epoch

$Loss : \mathcal{L}$, $Neural\ \ Network : NN$
for each sequence $S^{(i)}$ **do**
  $C^{(i)} \leftarrow$ **Context**$(S^{(i)})$
  for each step $S_t^{(i)}$ in $S^{(i)}$ **do**
    $\boldsymbol{\pi}_t^{(i)} \leftarrow$ **NN**$(C^{(i)}, S_0^{(i)}, \ldots, S_{t-1}^{(i)})$
    $L \mathrel{+}= L(S_t^{(i)}; \boldsymbol{\pi}_t^{(i)})$
  end for
end for
$NN \leftarrow \min(L, NN)$

Overall Loss Function. With the individual loss functions specified, the overall loss for the model is calculated as:

$$\mathcal{L} = \sum_i \sum_t \sum_{j=0}^{k-1} \mathcal{L}\left(S_t^{(i)}, \pi_{t,j}^{(i)}\right) \tag{1}$$

where k represents the total number of variables.

Training Epoch. A training epoch for our model, outlined in Algorithm 1, involves iterating over the sequences and time steps, updating the model parameters, and calculating the loss at each step.

For Algorithm 1, the loss function $\mathcal{L}$ is defined as (1), and the neural network is built with 4 layers: a GRU layer in between 2 dense layers, alongside with a final layer with different activation functions depending on the column data type. For more details, see section 3.1.2 of [33].

Next, for each sequence $S^{(i)}$, the constant input **Context** is assigned to the variable $C^{(i)}$. Then, for each row $S_t^{(i)}$, $C^{(i)}$ and all $S_0^{(i)}, \ldots, S_{t-1}^{(i)}$ are inputted into the neural network to output parameters $\pi_{(t,0)}^{(i)}, \pi_{(t,1)}^{(i)}, \ldots$. The training loss are then calculated from the parameters and added to the total loss. Finally, the loss function for the neural network is then minimized and outputted. For more details, see section 3.1 of [33].

3.3 Preprocessing Credit Card Transaction Dataset

We present a variety of preprocessing schemas to generate synthetic credit card transaction datasets via the CPAR model, each introducing unique modifications for comprehensive analysis. The initial dataset and metadata configurations serve as the foundation for each schema.

Schema 1: Minimal CPAR Requirements. To meet the minimum requirements for running the CPAR model, we focus on three users and transform the dataset as follows: combine `Year`, `Month`, `Day`, and `Time` into a single Pandas datetime column; convert `Amount` to a numerical format, removing the $ sign; replace NaNs in `Zip` with 'not applicable' for non-U.S. zip codes, converting valid zips to 'string'; change NaNs in `Errors?` to 'none', indicating error-free transactions; adjust `Merchant Name` to string, and `Is Fraud?` to boolean.

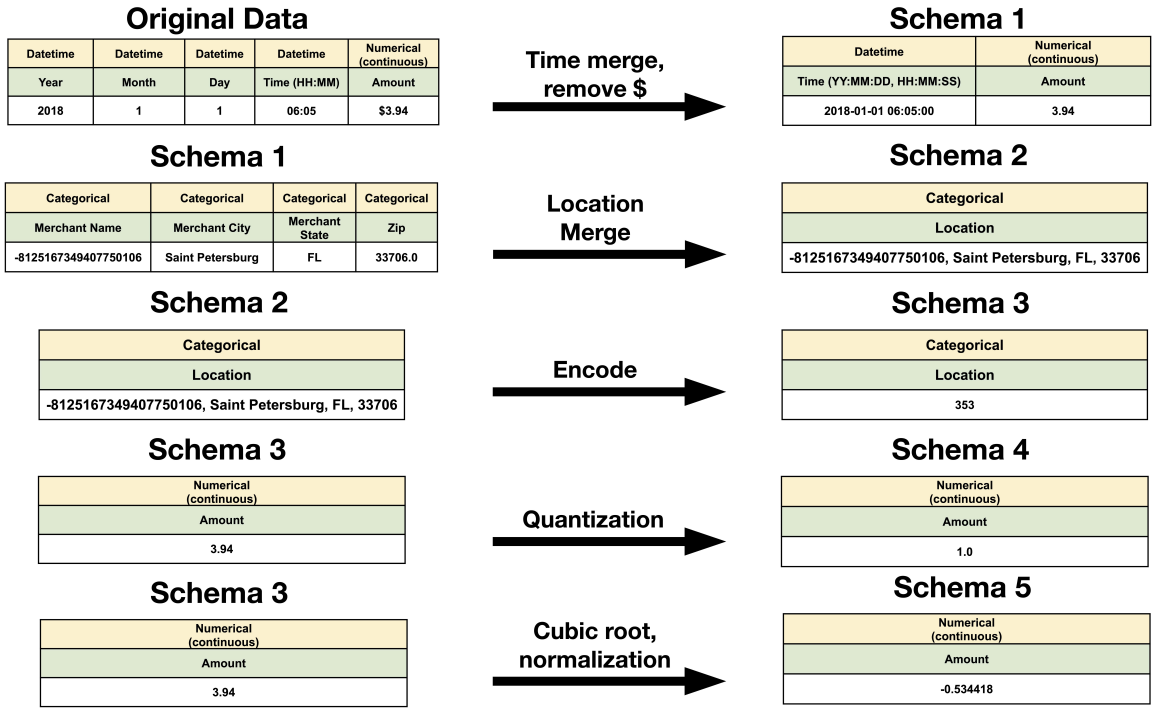

Figure 2: Differences of Columns between Schemas

With metadata from the preprocessed data, we apply CPAR (parameters: `epochs` = 1024, `context_columns` = []) to generate the initial synthetic dataset.

Schema 2: Handling Location and Avoiding Non-Existent Entries. Building on the previous schema, we integrate `Merchant Name`, `Merchant City`, `Merchant State`, and `Zip` into a single 'string' data type `Location` column to *prevent the creation of non-existent locations*, a common issue in Schema 1 due to possible shifts in these four columns. With updated metadata marking `Location` as 'categorical', we proceed to generate the second synthetic dataset using the same CPAR parameters.

Schema 3: Enhancing Machine Learning Efficacy through Categorical Encoding. Building upon prior schemas, we apply Scikit-Learn's Label Encoder to categorical columns (`Use Chip`, `MCC`, `Errors?`, `Location`, and `Is Fraud?`), maintaining their 'string' data types and unchanged metadata. This strategy ensures CPAR's accurate recognition of these as categorical, avoiding the potential generation of out-of-category values in the case of 'integer' data types. Post-encoding, we generate the third synthetic dataset and revert the columns to their original states using the trained encoders.

Schema 4: Addressing Non-Gaussianity with Quantile Transformation. Extending previous steps, Schema 4 employs quantile encoding for the `Amount` column, transforming its values logarithmically into 10 bins, following [20]. Post-transformation, this column is converted to 'string' to generate the fourth synthetic dataset, before undergoing an inverse transformation to revert changes.

Schema 5: Mitigating Non-Gaussianity with Cubic Root Transformation.

Building on Schema 3, this final schema standardizes the `Amount` column to zero mean and unit variance, followed by a cubic root transformation. This prepares the data for the fifth synthetic dataset generation, after which all transformations and scalings are reversed to preserve data integrity.

Each schema enhances the dataset's structure and content, facilitating a thorough analysis through the CPAR model.

3.4 Training Fraud Detection Model

This subsection outlines the process of training a fraud detection model, detailing each step from data preparation to performance evaluation.

- **Step 1: Synthetic Data Generation.** Utilizing schema 5, we generate synthetic datasets for 3 users with lengths determined by the CPAR model, choosing the first dataset for our machine learning efficacy experiment.
- **Step 2: Data Categorization.** We partition the datasets into fraud and non-fraud samples, subsequently using CPAR to create a synthetic dataset comprising 15,000 fraud transactions.
- **Step 3: Dataset Preparation.** Five datasets, each with 10,000 samples and varying fraud-to-non-fraud ratios (1%, 5%, 10%, 20%, and 50%), are prepared. The `Time` column is dropped, and a `time_diff` column is added to indicate *the time since the last event in minutes*, initialized to 0 for the first transaction of each user.
- **Step 4: Model Training.** We train models using the prepared datasets, employing metadata from 3.1, *context* = [], and setting epochs to 1024.
- **Step 5: Performance Evaluation.** Model performance is assessed by comparing accuracy (False Positive Rate and False Negative Rate) across the five datasets using original data as ground truth.

Through these steps, we conduct a thorough evaluation of our fraud detection models, ensuring that they are well-suited to handle the complexities and challenges of identifying fraudulent activities in transactional data. The use of synthetic data, coupled with careful data preparation and categorization, ensures that our models are trained on relevant and representative data, providing a solid foundation for accurate and reliable fraud detection.

Training on original and encoded data. For measuring machine learning efficacy, not only did we trained with untransformed data, but also did we consider encoded data. We label encoded every categorical column in the synthetic datasets and added identical `time_diff` column to the datasets. Then, we proceeded to train new machine learning models with these synethetic datasets, and compared the performance of machine learning models trained on encoded and un-encoded data.

Fraud Detection Model Class. To conduct a comprehensive efficacy analysis, we have implemented three distinct fraud detection models, each designed to handle varying fraud-to-non-fraud ratios. These models are selected for their unique strengths and capabilities in tackling different aspects of fraud detection, ensuring a robust and versatile evaluation.

(I) Categorical Boosting (Catboost) [8] CatBoost is a powerful algorithm renowned for its proficiency in managing categorical features directly without necessitating preprocessing, which can be a game changer in fraud detection where data is often diverse and complex. In our setup, we specify the 'categorical_feature_indices' parameter to ensure that the model accurately recognizes and utilizes the categorical data within the dataset. We follow a standard 80%-20% train-test split to assess the model's performance and prevent overfitting. In terms of hyperparameters, we have settled on 500 iterations, a learning rate of 0.01, and a tree depth of 5, ensuring a balanced trade-off between model complexity and training efficiency. The 'Logloss' loss function is employed to optimize the model's performance, and we set a random seed of 200 to guarantee reproducibility in our results. The model's effectiveness is meticulously validated using the original dataset to ensure authenticity and reliability in its fraud detection capabilities.

(II) Extreme Gradient Boosting (XGBoost) [4] XGBoost stands out for its scalability and computational efficiency, which is paramount in fraud detection due to the high volume of transactions that need to be analyzed promptly. We harness the power of the 'hist' tree method in XGBoost, which is known for its faster computation times and efficient memory usage, coupled with its ability to handle categorical inputs seamlessly. After splitting our data following the 80%-20% rule for training and testing, we configure the model with hyperparameters akin to those used in CatBoost, albeit with 'binary logistic' specified as the objective function to tailor the model for binary classification tasks, which is a common scenario in fraud detection. A different random seed is set to ensure diversity and robustness in our results. As with the previous model, we use the original data to validate the model's performance, ensuring a thorough and accurate evaluation.

(III)Light Gradient Boosting Machine (LGBM) [14] LGBM is renowned for its efficiency and speed, making it an ideal candidate for fraud detection tasks that demand quick and accurate results. It utilizes Gradient-Based Decision Trees (GBDT), Gradient-based One-Side Sampling (GOSS), and Exclusive Feature Bundling (EFB) to enhance its performance and efficiency. Following the standard practice, we split our dataset into 80% training and 20% testing portions. The model is then configured with 500 iterations, a learning rate of 0.01, and a 'binary' objective to align with the nature of fraud detection tasks. The 'binary_logloss' metric is used to assess the model's performance, ensuring a precise evaluation. A random state of 100 is set for reproducibility. Similar to the other models, we validate LGBM's performance using the original dataset, ensuring a reliable and accurate assessment of its fraud detection capabilities.

By leveraging the strengths of CatBoost, XGBoost, and LGBM, we aim to provide a comprehensive analysis that addresses various challenges in fraud detection, ultimately contributing to more secure and trustworthy financial transactions. Each model's unique features are meticulously harnessed to optimize their performance in fraud detection, providing us with valuable insights and a robust evaluation of their effectiveness.

4 Evaluation Results

In this section, we give empirical evaluation on the fidelity and utility of resulting synthetic credit car transaction time series. At section 4.1, we present fidelity evaluation on categor-

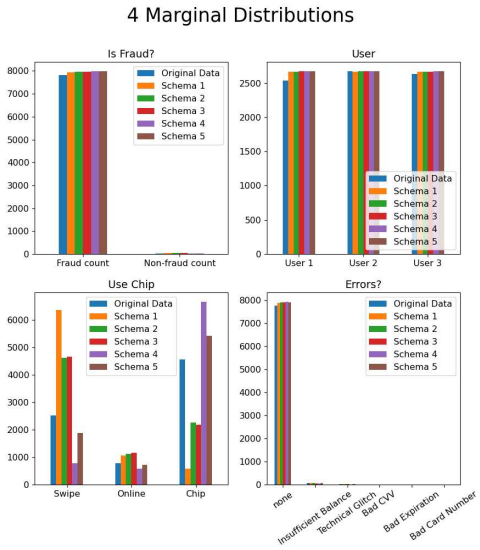

Figure 3: Marginal Distributions of columns `Is Fraud?`, `User`, `Use Chip`, `Errors?`, from original data set and Schema 1 to 5

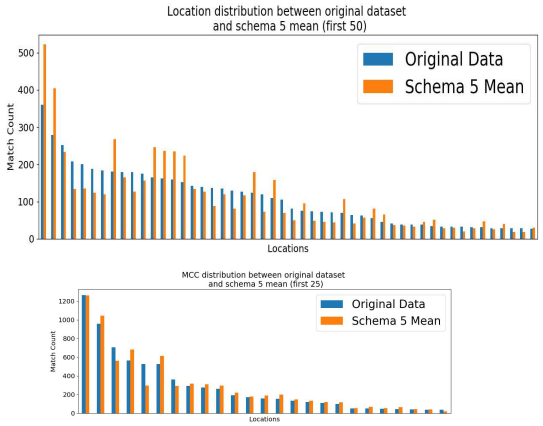

Figure 4: Marginal Distribution of Location (first 50 entries) for Synthetic dataset, Schema 5

ical variable (`MCC Errors?`, `Users`, `Use Chip`, `Location`, `Is Fraud?`). At section 4.2, we present fidelity evaluation on continuous variable (`Amount`). At section 4.3, we present utility evaluation on the fraud detection model trained on the synthetic credit card transaction dataset.

4.1 Fidelity of Synthetic Categorical Variable

Figure 3, first plot: This plot shows 4 marginal distributions of low-cardinal categorical columns (`Is Fraud?`, `User`, `Use Chip`, `Errors?`). Specifically, the plots display the counts of each category across the original dataset and all synthetic dataset generated from each schema. We also note that the column `Card` is not included, as it remains a constant for all datasets. The first plot in figure 3 indicates that for columns `Is Fraud?`, `User`, and `Errors?`, the differences in marginal distributions between the ground truth and all 5 datasets generated from each schema are minimal. For column `Use Chip`, however, the plot indicates that datasets generated from schema 1, 2, 3, and 4 (colors orange, green, red, and purple, respectively) are visibly farther from the original dataset (blue) than the dataset generated from schema 5 (brown). Thus, **most synthetic datasets closely match the original, except for "Use Chip".**

Figure 4, first plot: For the marginal distribution of `Location`, we first sort the count of unique locations in the original dataset in descending order, then plot out the first 50 comparisons with the mean count across all 20 Schema 5. The second plot in figure 2 indicates that for locations that have higher counts in the original dataset, there exists some discrepancies between the original dataset and the dataset generated from schema 5. Moving to the right of the plot where locations have lower counts, the discrepancies starts to decrease, and the distribution is more similar. Thus, we found **discrepancies between**

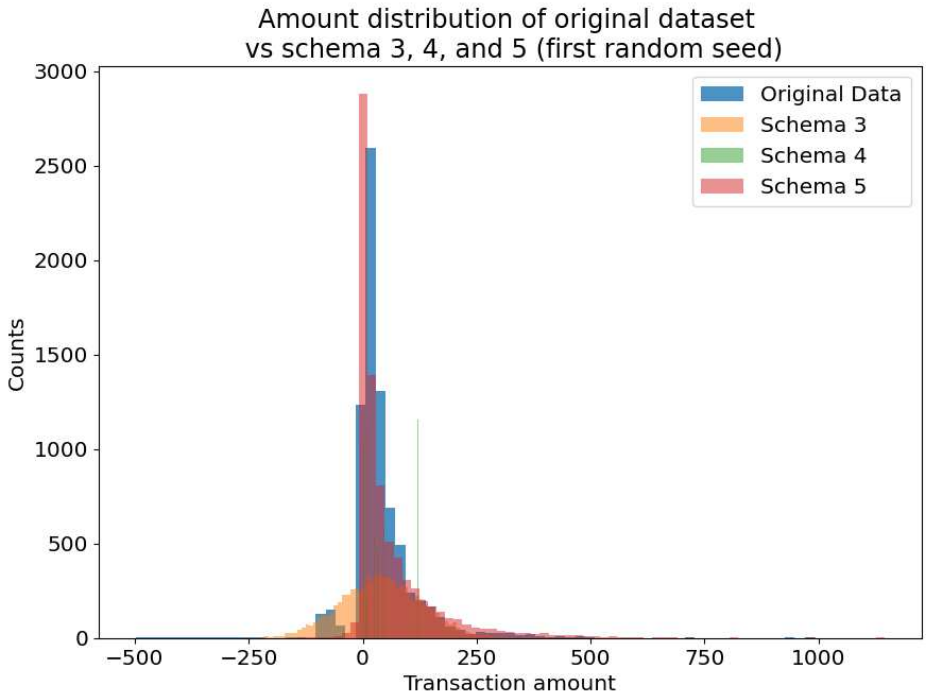

Figure 5: Amount Distribution for Original Dataset vs Synthetic Datasets Generated From Schema 3, 4, and 5

original and schema 5 datasets decrease for lower-count locations.

Figure 4, second plot: For the marginal distribution of `MCC`, we also sort the count of unique MCC in the original dataset in descending order. Then, we plot out the first 25 comparisons with the mean count across all 20 Schema 5. The third plot in figure 2 indicates that the marginal distribution of MCC between the original dataset and the dataset generated from schema 5 are really close to each other. Thus, we found **The MCC distribution in Schema 5 closely matches the original.**

4.2 Fidelity of Synthetic Continuous Variable

Figure 5 gives the fidelity evaluation result for the continuous variable in the credit card transaction dataset. The figure indicates that dataset generated from schema 3 (before data transformation, in orange), shows a shifted Gaussian distribution, which is accurate from the description of the CPAR model. The dataset generated from schema 4 (green) shows multiple long and thin bars as a result of transaction amount quantization. The dataset generated from schema 5 (red) shows a distribution that is really close with the marginal distribution of the transaction amounts in the original dataset. The plot indicates that between the 3 datasets generated from schema 3, 4, and 5, schema 5 is the closest to the original data by a large margin. Thus, we found **Schema 5 closely matches the original transaction amount distribution.**

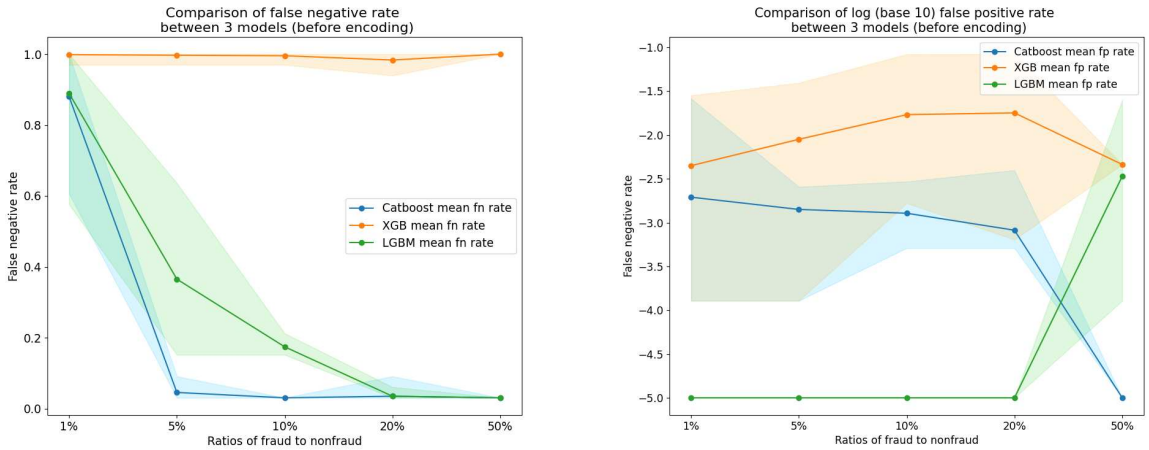

Figure 6: False Negative Rate (Left) and False Positive Rate (Right) for Fraud Detection Result of `Catboost, XGBoost` and `LGBM`.

4.3 Utility of Synthetic Data trained Fraud Detection model

Figure 6 gives a utility evaluation of synthetic data trained fraud detection model. Note that since the original dataset is highly imbalanced (0.4206% fraud transactions and 99.5794% non-fraud transactions), monitoring the false positive and false negative rate is highly important, as high accuracy does not imply low false positive or false negative rates.

The left figure indicates that for the false negative rate across all 3 machine learning models, Catboost and LGBM have the best resutls, with Catboost reaching and LGBM reaching 3.0303% in mean false negative rate across training on 20 different seeds (essentially only missing 1 out of 33 fraud transactions) at 20% and 50% fraud to non-fraud ratio. While LGBM suffers from higher uncertainty, we note that training speed of LGBM is way faster than the training speed of Catboost. On the other hand, XGBoost fails to deliver great result, with a mean of almost 100% across all fraud to non-fraud ratios. This might be caused by the lack of support of categorical features in the library (the support for categorical features is currently still in experimentation). Thus, we found **`Catboost` and `LGBM` outperform `XGBoost` in false negative rate.**

The right figure indicates that for the log base 10 of the false positive rate across all 3 machine learning models, the false positive rate of Catboost continues to decrease (at last to ¡0.0001%) as fraud to non-fraud ratio increases, while LGBM encounters a great leap in false positive ratio when moving from 20% fraud to non-fraud ratio to 50% fraud to non-fraud ratio. XGBoost, on the other hand, maintains a high false positive rate throughout all the different fraud to non-fraud ratios. Thus, we found **`Catboost` outperform `LGBM` and `XGBoost` in false positive rate.**

Lastly, note that the previous models are trained from decoded data, meaning that they are similar to that of the original data. We also trained our models using encoded **Dataset: Schema 5** and validated on encoded original data, which is the same encoded schemas as **Dataset: Schema 3 and 5**. The results we discovered is that for mean false negative rate across all fraud to non-fraud ratios, all models remain robust (cite appendix). For mean false positive rate, on the other hand, both XGBoost and Catboost remain robust, while Light-

GBM appears to be sensitive to label encoding.Thus, we found `Light-GBM` **is sensitive to label encoding, while** `XGBoost` **and** `Catboost` **remain robust.**

5 Conclusion

We evaluate the fidelity and utility of synthetic data generated from the application of the CPAR model to a real-world Credit Card Transaction dataset. By employing Schema 5 for preprocessing the original dataset, we ensure that the distribution of both categorical and continuous variables is well-preserved in the synthesized time-series data. Based on Schema 5, we generate synthetic data to train three representative fraud detection models (`Catboost, XGBoost, LGBM`), further investigating the utility of the synthetic data as training material for machine learning. Utilizing the synthetic data allows us to increase the proportion of fraud cases in our training dataset, leading to near-zero False Positive and False Negative Rates for `Catboost` and `LGBM`. From the insights gained in this study, we posit that, **given the appropriate preprocessing schema, synthetic data can indeed serve as a high-fidelity copy of the original data, enhancing the performance of fraud detection by generating additional fraud case data.**

References

[1] Erik Altman, Béni Egressy, Jovan Blanuvsa, and Kubilay Atasu. Realistic synthetic financial transactions for anti-money laundering models. *ArXiv*, abs/2306.16424, 2023.

[2] Erik R. Altman. Synthesizing credit card transactions. *Proceedings of the Second ACM International Conference on AI in Finance*, 2019.

[3] Vadim Borisov, Kathrin Seßler, Tobias Leemann, Martin Pawelczyk, and Gjergji Kasneci. Language models are realistic tabular data generators. *ArXiv*, abs/2210.06280, 2022.

[4] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.

[5] Yinan Cheng, Chi-Hua Wang, Vamsi K. Potluru, Tucker Balch, and Guang Cheng. Downstream task-oriented generative model selections on synthetic data training for fraud detection models. In *ACM International Conference on AI in Finance – Workshop*, November 5 2022.

[6] Kyunghyun Cho, Bart van Merrienboer, Caglar Gülcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Conference on Empirical Methods in Natural Language Processing*, 2014.

[7] Abhyuday Desai, Cynthia Freeman, Zuhui Wang, and Ian Beaver. Timevae: A variational auto-encoder for multivariate time series generation. *ArXiv*, abs/2111.08095, 2021.

[8] Anna Veronika Dorogush, Vasily Ershov, and Andrey Gulin. Catboost: gradient boosting with categorical features support. *ArXiv*, abs/1810.11363, 2018.

[9] Khaled El Emam, Lucy Mosquera, and Richard Hoptroff. *Practical synthetic data generation: balancing privacy and the broad availability of data*. O'Reilly Media, 2020.

[10] European Commission. 2018 reform of eu data protection rules. 2018.

[11] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232, 2001.

[12] Yu. V. Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. Revisiting deep learning models for tabular data. In *Neural Information Processing Systems*, 2021.

[13] James Jordon, Lukasz Szpruch, Florimond Houssiau, Mirko Bottarelli, Giovanni Cherubin, Carsten Maple, Samuel N Cohen, and Adrian Weller. Synthetic data–what, why and how? *arXiv preprint arXiv:2205.03257*, 2022.

[14] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In *Neural Information Processing Systems*, 2017.

[15] Jayoung Kim, Chae Eun Lee, and Noseong Park. Stasy: Score-based tabular data synthesis. *ArXiv*, abs/2210.04018, 2022.

[16] Akim Kotelnikov, Dmitry Baranchuk, Ivan Rubachev, and Artem Babenko. Tabddpm: Modelling tabular data with diffusion models. *ArXiv*, abs/2209.15421, 2022.

[17] Jaehoon Lee. Invertible tabular gans: Killing two birds with onestone for tabular data synthesis. In *Neural Information Processing Systems*, 2022.

[18] Flavio Di Martino and Franca Delmastro. Explainable ai for clinical and remote health applications: a survey on tabular and time series data. *Artificial Intelligence Review*, 56:5261 – 5315, 2022.

[19] Krishna Modi and Reshma Dayma. Review on fraud detection methods in credit card transactions. *2017 International Conference on Intelligent Computing and Control (I2C2)*, pages 1–5, 2017.

[20] Inkit Padhi, Yair Schiff, Igor Melnyk, Mattia Rigotti, Youssef Mroueh, Pierre L. Dognin, Jerret Ross, Ravi Nair, and Erik Altman. Tabular transformers for modeling multivariate time series. *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3565–3569, 2020.

[21] Neha Patki, Roy Wedge, and Kalyan Veeramachaneni. The synthetic data vault. *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 399–410, 2016.

[22] Karthikeswaren Ramachandran, Kanishka Kayathwal, Hardik Wadhwa, and Gaurav Dhama. Fraudammo: Large scale synthetic transactional dataset for payment fraud detection. *2023 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7, 2023.

[23] Prajit Ramachandran, Barret Zoph, and Quoc V. Le. Searching for activation functions. *ArXiv*, abs/1710.05941, 2018.

[24] Aaditya Ramdas, Nicolás García Trillos, and Marco Cuturi. On wasserstein two-sample testing and related families of nonparametric tests. *Entropy*, 19:47, 2015.

[25] Abhimanyu Roy, Jingyi Sun, Robert Mahoney, Loreto Peter Alonzi, Stephen Adams, and Peter A. Beling. Deep learning detecting fraud in credit card transactions. *2018 Systems and Information Engineering Design Symposium (SIEDS)*, pages 129–134, 2018.

[26] Aivin V. Solatorio and Olivier Dupriez. Realtabformer: Generating realistic relational and tabular data using transformers. *ArXiv*, abs/2302.02041, 2023.

[27] State of California Department of Justice. 2018 california consumer privacy act. 2018.

[28] Belén Vega-Márquez, Cristina Rubio-Escudero, José Cristóbal Riquelme Santos, and Isabel A. Nepomuceno-Chamorro. Creation of synthetic data with conditional generative adversarial networks. In *Soft Computing Models in Industrial and Environmental Applications*, 2019.

[29] Giorgio Visani, Giacomo Graffi, Mattia Alfero, Enrico Bagli, Davide Capuzzo, and Federico Chesani. Enabling synthetic data adoption in regulated domains. *arXiv preprint arXiv:2204.06297*, 2022.

[30] Philip B. Weerakody, Kevin Kok Wai Wong, Guanjin Wang, and Wendell Ela. A review of irregular time series data handling with gated recurrent neural networks. *Neurocomputing*, 441:161–178, 2021.

[31] Lei Xu and Kalyan Veeramachaneni. Synthesizing tabular data using generative adversarial networks. *ArXiv*, abs/1811.11264, 2018.

[32] Hongyi Yuan, Songchi Zhou, and Sheng Yu. Ehrdiff: Exploring realistic ehr synthesis with diffusion models. *ArXiv*, abs/2303.05656, 2023.

[33] Kevin Alex Zhang, Neha Patki, and Kalyan Veeramachaneni. Sequential models in the synthetic data vault. *ArXiv*, abs/2207.14406, 2022.

[34] Zilong Zhao, Robert Birke, and Lydia Yiyu Chen. Fct-gan: Enhancing table synthesis via fourier transform. *ArXiv*, abs/2210.06239, 2022.

[35] Zilong Zhao, Aditya Kunar, Robert Birke, and Lydia Yiyu Chen. Ctab-gan+: Enhancing tabular data synthesis. *ArXiv*, abs/2204.00401, 2022.

[36] Zilong Zhao, Aditya Kunar, Hiek van der Scheer, Robert Birke, and Lydia Yiyu Chen. Ctab-gan: Effective table data synthesizing. *ArXiv*, abs/2102.08369, 2021.

A Appendix

This section provides additional empirical results for Implementation of `CPAR` Model On Credit Card Transaction Data discussed at Section 4.

Metric-Oriented results.

- Figure 7 gives the first 300 entries of the relationship between time since each event (transactions) and the transaction amounts.
- Figure 10 gives the false negative and false positive rate of the machine learning models trained using encoded synthetic credit card transaction data.
- Figure 11 gives the 51th to 150th entries of the marginal distribution of `Location` between original data and synthetic data from Schema 5.
- Figure 12 gives the 26th to 75th entries of the marginal distribution of `MCC` between original data and synthetic data from Schema 5.

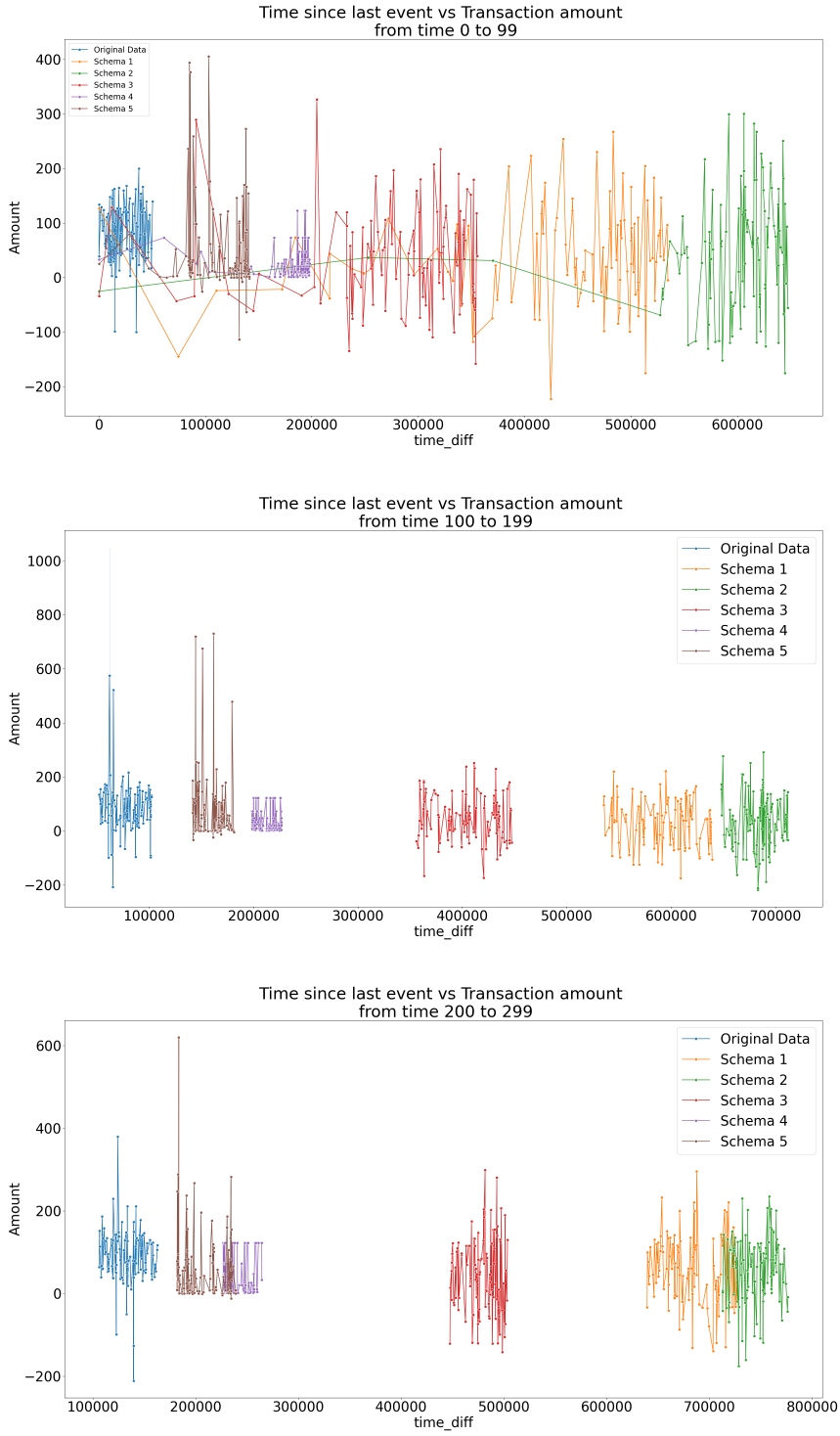

Figure 7: Time Since Last Event vs Transaction Amount (Time Dependency). Even though there are obvious gaps between the time dependency graph of the original data (in blue) to the data from Schema 1 to 5, the distance between the original and Schema 5 (in brown) is the lowest.

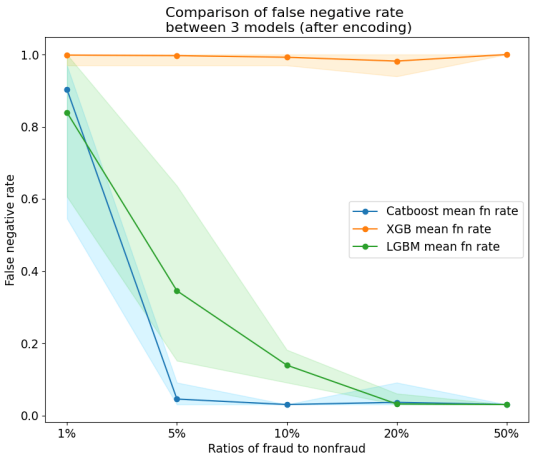

Figure 8: first figure

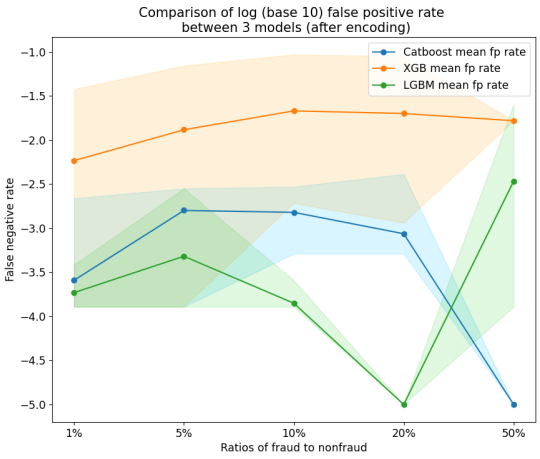

Figure 9: second figure

Figure 10: FNR and FPR. `Light-GBM` is found to be sensitive to label encoding, while `XGBoost` and `Catboost` remain robust.

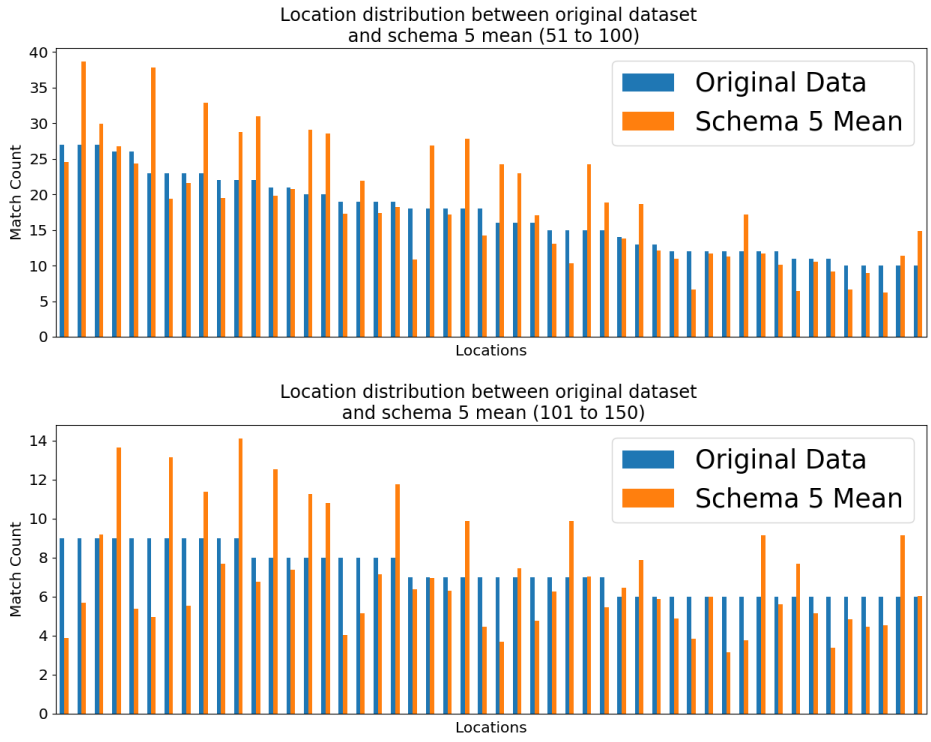

Figure 11: Marginal Distribution of the `Location` column. The next 100 entries are shown, where after the 150th entry the single digit matching counts become insignificant.

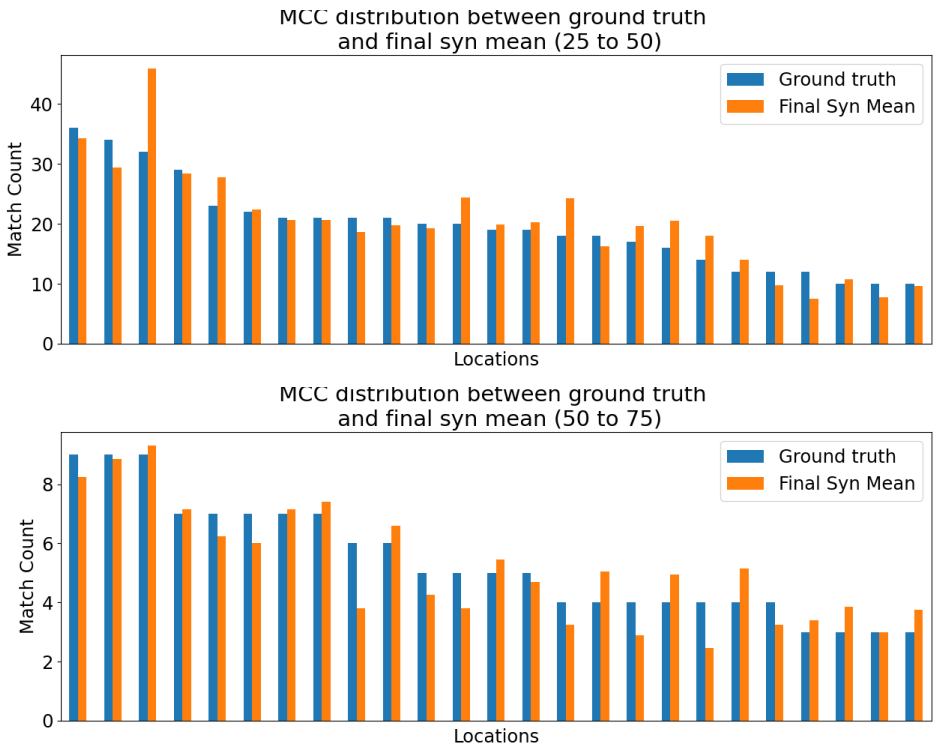

Figure 12: Marginal Distribution of the `MCC` column. The next 50 entries are shown, and, again, where after the 75th entry the single digit matching counts become insignificant.